On Measures to Build Linkage Trees in LTGA

Peter A.N. Bosman[1] and Dirk Thierens[2]

[1] Centre for Mathematics and Computer Science, P.O. Box 94079,
1090 GB Amsterdam, The Netherlands
Peter.Bosman@cwi.nl
[2] Department of Information and Computing Sciences, Utrecht University,
Utrecht, The Netherlands
D.Thierens@uu.nl

Abstract. For an evolutionary algorithm (EA) to be efficiently scalable, variation must be linkage friendly. For this reason many EAs have been introduced that build and exploit linkage models, amongst which are estimation-of-distribution algorithms (EDAs). Although various models have been empirically evaluated, it remains of key importance to better understand the conditions under which model building is successful. In this paper, we consider the linkage tree genetic algorithm (LTGA). LTGA is a recent powerful linkage-learning EA that builds a hierarchical linkage model known as the linkage tree (LT). LTGA exploits this model using an intensive mixing procedure aimed at optimally exchanging building blocks. Empirical evaluation studies of LTGA have appeared in literature using different entropy-based measures for building the LT, but with comparable results. We study the differences in these measures to better understand the requirements for detecting important linkage information and point out why some measures are more successful than others.

1 Introduction

Having a tunable model that drives variation in an evolutionary algorithm (EA) potentially allows efficiently tackling a large class of optimization problems. Key to successfully solving a particular problem is the ability to configure this model for that problem, i.e. such that combining solutions leads to (significant) improvements. Key questions are whether such a proper configuration can be learned efficiently and what type of model and learning algorithm are required. In this paper we consider this question in the light of one of the latest and most promising model-building EAs for discrete optimization problems: the linkage tree genetic algorithm (LTGA) [1,5,10].

The type of EA that is perhaps best known for building and using models is the estimation-of-distribution algorithm (EDA). Models in EDAs represent probability distributions over the space of solutions. In EDAs, linkage information, i.e. which variables should be considered jointly when generating new solutions, is processed via probabilistic dependency relations. Although probability theory provides very powerful tools, estimating complete distributions might be more than what is required. Instead therefore, LTGA learns linkage relations directly,

C.A. Coello Coello et al. (Eds.): PPSN 2012, Part I, LNCS 7491, pp. 276–285, 2012.

although the statistical techniques used to do so have much in common with building probabilistic models as is done in EDAs.

LTGA exhibits excellent performance on several benchmark problems [1,5,10]. The measures used to learn the linkage model in these studies are however different, although all are entropy-based. In this paper we take a closer look at why the results using different measures are so similar in order to better understand the requirements for detecting important linkage information. To do so, we consider what it is we require of a linkage measure from a viewpoint of EA dynamics rather than from probability theory and notions of probabilistic independence as in EDAs.

2 The Linkage Tree Genetic Algorithm (LTGA)

Here we only briefly describe the most recent version of LTGA [1]. For more details we refer the interested reader to the related literature [1,5,10].

To model linkage, a linkage tree (LT) is used in which variables can be linked on one level but not linked on another, lower, level. At the lowest level, all variables are unlinked and form singleton sets. An LT then can be formed by merging pairs of sets until all sets are merged. An example of an LT is given in Figure 1.

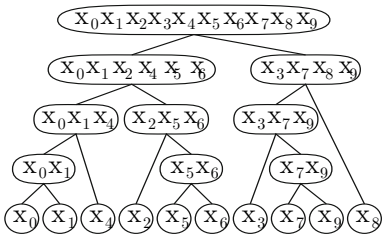


Fig. 1. Example of a LT for 10 variables

In each generation of LTGA a set of n solutions is selected from a population of size n using tournament selection with a tournament size of 2. The LT is learned from this set. New solutions are however generated from the population directly. Instead of fully creating new solutions first and only then evaluating fitness, LTGA uses a procedure called Genepool Optimal Mixing (GOM) [10]. For each solution in the population, exactly one offspring solution is generated. To do so, the solution is first cloned. The LT is then traversed in reverse-merging order, i.e. ending with all variables in separate groups, and skipping the top level. For each group, a donor solution is chosen randomly from the population. The values in the donor pertaining to the variables in the linkage group are then copied. If the solution is thereby improved, it is kept, otherwise the changes are reverted. The use of GOM increases selection pressure in a building-block-wise manner instead of on an entire solution basis. It is mostly because of OM that LTGA requires only very small population sizes compared to most EDAs.

To ensure efficient convergence to a single solution, in the latest version of LTGA [1] GOM is extended with forced improvements (FI). In FI, if a solution could not be improved by GOM, an additional GOM operation is performed on that solution but now with the currently known best solution as the donor, this time stopping as soon as an improvement is detected. FI introduces a special directed convergence pressure, through linkage space, toward the best solution found so far. Note that FI doesn't continuously reduce population diversity, but only if a solution couldn't be improved anymore anyway.

3 Measures to Build Linkage Trees

To build a LT, a similarity measure to be maximized or distance measure to be minimized is required in order to decide which two groups of variables to merge next. Building a LT is also known as hierarchical clustering [4].

3.1 Commonly Used Measures for Hierarchical Clustering

The most commonly adopted measures are based on mutual information (MI) and variation of information (VI). Both measures themselves are based on entropy. For a set X of random variables, the entropy $H(X)$ is given by:

$$H(X) = \sum_{x \in \Omega_X} -P(X = x)\log(P(X = x)) \quad (1)$$

where Ω_X is the sample space of X, i.e. all 2^k bit combinations for k binary variables. For two sets of random variables X and Y, MI and VI are given by:

$$MI(X,Y) = H(X) + H(Y) - H(X \cup Y) \quad (2)$$

$$VI(X,Y) = H(X \cup Y) - MI(X,Y) = 2H(X \cup Y) - H(X) - H(Y) \quad (3)$$

MI is a similarity measure whereas VI is a distance measure. MI is closely related to probabilistic model building in EDAs. Specifically, MI times the size of the data set is identical to the negative log-likelihood difference of two distributions that differ only in modelling X and Y independently or jointly. This difference is part of extended-likelihood measures, which are commonly used, e.g. in the well-known EDA ECGA [3].

The actual measures commonly encountered in hierarchical clustering literature are normalized versions of MI and VI. The reason for this is that the range of both measures is dependent on the number of variables in X and Y. This results in a bias to favoring large sets when deciding which two sets to merge [4]. Different normalizations are possible. We denote the normalized version used in LTGA by NVI. We similarly normalize MI and denote that by MNI, giving:

$$MNI(X,Y) = MI(X,Y)/H(X \cup Y) = (H(X) + H(Y))/H(X \cup Y) - 1 \quad (4)$$

$$NVI(X,Y) = VI(X,Y)/H(X \cup Y) = 2 - (H(X) + H(Y))/H(X \cup Y) \quad (5)$$

Although the direct use of NVI results in well-balanced LTs in LTGA and excellent optimization performance of LTGA [9], joint entropies need to be computed for every candidate set. High up in the LT these sets contain many variables. This poses a computational burden because Equation 1 requires summing over all variable configurations (encountered in the population). For this reason, a measure adaptation known as UPGMA (unweighted pair group method with arithmetic mean) was used in recent versions of LTGA [1,5,10]. With UPGMA all possible pairs of variables are considered. This is computationally beneficial if the computational effort to compute a measure grows faster than quadratic

in the number of variables, which is the case for VI (and MI). The UPGMA adaptation of a measure M is given by:

$$M^{UPGMA}(X,Y) = \frac{1}{|X||Y|} \sum_{X_i \in X} \sum_{Y_j \in Y} M(\{X_i\},\{Y_j\}) \tag{6}$$

LTGA with an UPGMA adaptation of NVI performs at least as good as LTGA with NVI, in terms of the required number of function evaluations [5]. However, with UPGMA, every actual VI computation is performed for just 2 random variables. Arguably therefore normalization is no longer required. For this reason LTGA was also recently tested with UPGMA and MI, obtaining apparently comparable results [10]. This raises the natural question of how these measures really guide hierarchical clustering and thereby LTGA.

3.2 Measures from a Viewpoint of EA Dynamics

What we ideally desire from a EA is efficient mixing of building blocks, i.e. instances of sets of variables that have an above-average contribution to a solutions' quality. Selection gives these building blocks more copies, thereby reducing the diversity of instances for the involved variables. The latter is exactly what is measured by H (Equation 1). This therefore suggests that we could minimize $H(X,Y) = H(X \cup Y)$ when deciding which sets of variables X and Y to merge.

However, reducing the dispersion of instances by itself isn't sufficient because this also happens simply because the EA converges. Thus, from a viewpoint of EA dynamics H has an undesirable bias toward more converged variables. This can also be seen in Figure 2. Combinations with the converged variable X_1 lead to a lower H than when variables X_0 and X_2 are combined, merely because the entropy of X_1 itself is 0. Instead therefore, what is really of interest is the *change* in dispersion of instances when going from possible combinations of available instances of X and Y to actually available instances of $X \cup Y$. Such an effect is exactly what MI (Equation 2) measures. Indeed, in Figure 2 we see that when considering what variable to best join X_0 with, X_2 is best when using MI whereas using H directly the converged and uninteresting variable X_1 appears best.

Data	X_0	X_1	X_2
0	1	1	0
1	1	1	1
2	0	1	0
3	1	1	1
4	1	1	0
5	1	1	1
6	1	1	1
7	0	1	0
8	0	1	0
9	1	1	1

H	X_0	X_1	X_2
	0.88	0.00	1.00

MI	X_0	X_1	X_2
X_0	0.88	0.00	0.40
X_1	0.00	0.00	0.00
X_2	0.40	0.00	1.00

NMI	X_0	X_1	X_2
X_0	1.00	0.00	0.27
X_1	0.00	1.00	0.00
X_2	0.27	0.00	1.00

H	X_0	X_1	X_2
X_0	0.88	0.88	1.48
X_1	0.88	0.00	1.00
X_2	1.48	1.00	1.00

VI	X_0	X_1	X_2
X_0	0.00	0.88	1.09
X_1	0.88	0.00	1.00
X_2	1.09	1.00	0.00

NVI	X_0	X_1	X_2
X_0	0.00	1.00	0.73
X_1	1.00	0.00	1.00
X_2	0.73	1.00	0.00

Fig. 2. Example of all measures for a specific set of data and 3 random variables

Since VI is a negation of MI, it appears equally useful. However, the negation involves $H(X,Y)$, resulting in $H(X,Y)$ weighing twice as heavy in VI as it does in MI (Equations 2 and 3). As a result, the bias in H toward more converged variables rings through more in VI. Considering EA dynamics, VI is therefore further away from the desirable properties of a linkage detection measure than MI. Accordingly, using VI in Figure 2 results in a different preference relation than using MI. Now X_1 is best to combine with X_0, just like when using H.

As mentioned earlier, normalization removes the bias of VI to large clusters. The above however suggests VI additionally has a bias toward more converged variables. However, recent literature suggests that, in combination with UPGMA, LTGA using NVI [5] performs equally good as LTGA using MI [10]. Normalization thus appears to change *more* things. This can indeed already be seen in the example in Figure 2 because, similar to MI, when using NVI we find that the best variable to combine with X_0 is X_2. When we look closer, we can indeed see that normalization brings VI very close to MI. Clearly, from Equations 4 and 5 we have $NVI(X,Y) = 1 - MNI(X,Y)$. This also holds using UPGMA, since $NVI^{UPGMA}(X,Y) = (|X||Y|)^{-1} \sum_{X_i \in X} \sum_{Y_j \in Y} 1 - MNI(\{X_i\},\{Y_j\}) = (|X||Y|)^{-1}(|X||Y| - \sum_{X_i \in X} \sum_{Y_j \in Y} MNI(\{X_i\},\{Y_j\})) = 1 - MNI^{UPGMA}(X,Y)$. Using MNI or NVI to build an LT is therefore an identical approach. Contrary to the use of H and VI, the use of NVI is however quite similar to the use of MI, as can be seen by equivalently comparing MNI and MI. Normalization of MI shifts the importance of the *absolute* difference between joint entropy and the individual entropies to their *relative* difference. For EA dynamics this means that normalization brings the advantage that if variables are nearly converged, the reduction in instance dispersion that we want to detect can still lead to large "signals" for the learning algorithm to pick up. Compared to MNI, MI therefore has a larger preference for less converged variables. Note that this relative bias only plays a role in cases where a desirable reduction in instance dispersion is already present. Thus, this bias in MI toward less converged variables is likely not harmful like the bias in VI or in H toward more converged variables. Furthermore, it is beforehand unclear whether MI or MNI is to be preferred in LTGA. Preferring entropy reductions in less converged variables (i.e. MI) allows to reduce noise in the overall optimization process faster. The opposite however allows first considering variables whose diversity is running out the fastest.

For all combinations of measures we empirically determined how often they disagree. For a fine-grained sampling of all possible probabilities for two sets of two binary variables $\{X_0, X_1\}$ and $\{X_2, X_3\}$ we determined, for two different measures M_0 and M_1, whether $M_i(\{X_0\},\{X_1\}) \succ M_i(\{X_2\},\{X_3\})$, $i \in \{0,1\}$. We also created correlation graphs for measure differences $M_i(\{X_2\},\{X_3\}) - M_i(\{X_0\},\{X_1\})$. The results are shown in Figure 3. Because MI and MNI need to be maximized and H, VI and NVI need to be minimized, we replaced MI and MNI by -MI and -MNI. In the correlation graphs, two measures agree if the observed differences are both positive or both negative (i.e. all-positive and all-negative quadrants). Moreover, observations for H and VI were divided by 2 to ensure that all measures have a difference in the range $[-1;1]$.

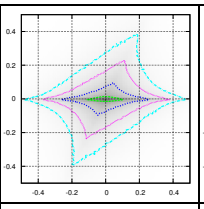
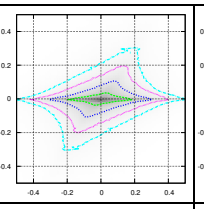
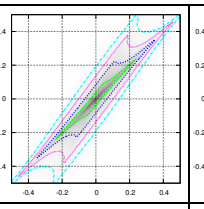
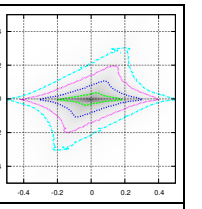
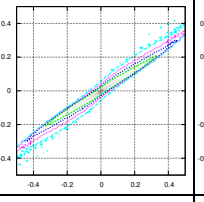
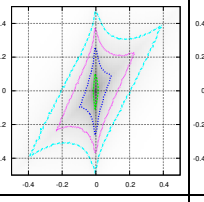
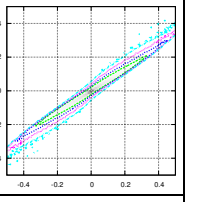
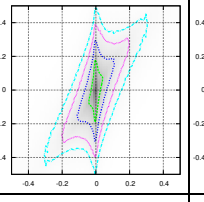
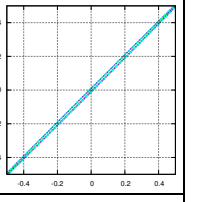
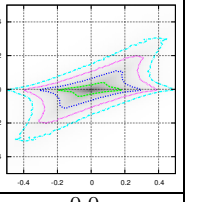
	H	MI	MNI	VI	NVI
H	0.0				
MI	40.4	0.0			
MNI	37.0	3.4	0.0		
VI	10.1	30.3	26.9	0.0	
NVI	37.0	3.4	0.0	26.9	0.0

Fig. 3. Percentage of cases in which measures disagree about the preference ordering of two sets of random variables (lower-left triangle) and gray-value coded maps of the density of observed combinations of measure differences for two sets of random variables, overlaid with density contours

Figure 3 shows a strong agreement of VI with H. Given the strong bias of H toward more converged variables, VI is, like H, expected to lead to less efficient behavior of LTGA. MI is decorrelated the most with H, followed by MNI. This is in accordance with our earlier finding that MI is relatively more biased to less converged variables. NVI and MNI are, as expected, in perfect agreement. The difference between MI and MNI is small. When they do differ, the measures themselves exhibit only small differences (i.e. decisions are a "close" call).

Ultimately, it is the optimization performance of the EA that is of importance. From the analysis above it is to be expected that the use of MI and MNI (and, equivalently, NVI) in LTGA leads to better performance on non-randomly structured problems than the use of H and VI. However, the difference between MI and MNI, i.e. normalization, is subtle. Therefore, in the next Section we empirically determine whether normalization does something desirable in terms of EA dynamics by running LTGA on various optimization problems.

4 Experiments

4.1 Optimization Problems

We consider three well-known benchmark problems from linkage-learning literature and one well-known NP-hard problem, all of which need to be maximized. The first problem is onemax in which every variable is independent of the others:

$$f_{\text{Onemax}}(\boldsymbol{x}) = \sum_{i=0}^{l-1} x_i$$

The second problem is the mutually-exclusive, additively decomposable sum of the well-known order-k deceptive trap functions [2] with $k = 5$:

$$f_{\text{Trap5}}(\boldsymbol{x}) = \sum_{i=0}^{(l/k)-1} f^{\text{sub}}_{\text{Trap-k}}\left(\sum_{j=ki}^{ki+k-1} x_j\right) \quad \text{, with } f^{\text{sub}}_{\text{Trap-k}}(u) = \begin{cases} 1 & \text{if } u = k \\ \frac{k-1-u}{k} & \textit{otherwise} \end{cases}$$

It is commonly known that the linkage groups pertaining to the subfunctions need to be detected and processed in order for optimization to proceed efficiently.

The third problem is the nearest-neighbour overlapping, additively decomposable sum of a-priori randomly generated subfunctions of length k, which constitutes a NK-landscape [6]. We use $k = 5$ and the maximum overlap of 4, but without wraparound:

$$f_{\text{NK-S1}}(\boldsymbol{x}) = \sum_{i=0}^{l-k} f^{\text{sub}}_{\text{NK}}\left(\boldsymbol{x}_{(i,i+1,\ldots,i+k-1)}\right)$$

where $f^{\text{sub}}_{\text{NK}}\left(\boldsymbol{x}_{(i,i+1,\ldots,i+k-1)}\right)$ is an a-priori randomly chosen value in $[0;1]$.

The NP-hard problem we consider is weighted MAXCUT. It is defined given a weighted undirected graph with a set of l vertices $V = \{v_0, v_1, \ldots, v_{l-1}\}$, a set of edges E between the vertices, and a weight w_{ij} for each edge $(v_i, v_j) \in E$. The goal is to split V into two sets such that the combined weight of edges that are thereby cut, i.e. running between vertices in different sets, is maximized. By introducing a binary variable x_i for every vertex that indicates if vertex v_i is either in set 0 or set 1, the function to be optimized is therefore:

$$f_{\text{weighted MAXCUT}}(\boldsymbol{x}) = \sum_{(v_i,v_j)\in E} \begin{cases} w_{ij} \text{ if } x_i \neq x_j \\ 0 \quad \textit{otherwise} \end{cases}$$

We encode this problem straightforwardly using the x_i directly. Benchmarks problem instances of various types and sizes exist in literature, but because we want to perform a controlled scalability analysis, we generated our own instances. For problem sizes $l \in \{6, 12, 25, 50, 100\}$ we generated fully connected graphs with $\frac{1}{2}l(l-1)$ edges. To set the weights, we follow the approach by Rubinstein [8] to obtain interesting instances and choose them randomly following a β distribution with parameters $\alpha = 100$, $\beta = 1$ and scaled to the range of $[1;5]$. For each problem size, we generate 10 instances. The maximum problem

size was chosen such that the exact optimizer BIQMAC [7] could provide optimal solutions within reasonable time. Because MAXCUT is NP-hard, we will consider both obtaining the optimum as well as obtaining 95% of the optimum where we accounted for the average random value ARV of an instance by setting the actual target value for an instance to $ARV + 0.95(OPT - ARV)$. The ARV is determined empirically by averaging over many randomly sampled solutions.

4.2 Experimental Setup

We say that a problem is solved if at least 99 out of 100 independent runs converged to either the global optimum or to a predefined sufficiently close approximation. Moreover, instead of stopping when the target is reached, we run until convergence (all solutions are the same) because we feel this is more realistic in practice where the optimum is not known beforehand and outcomes are typically collected upon termination.

For NK-S1, we have generated 100 instances per problem size randomly. The 100 independent runs are performed on 100 different, but always the same 100, instances per problem size. For weighted MAXCUT however, every problem instance is considered separately as is more typical of combinatorial optimization literature. The reason for this is that specific instances may be easy or hard and their properties may be interesting to study separately. Therefore, for weighted MAXCUT 100 independent runs are performed per problem instance. Note that this alters the interpretation of the range of outcomes where we aggregate per problem size. For weighted MAXCUT these ranges will be much larger.

A key performance indicator is scalability, i.e. how do the required resources (population size n and number of required evaluations) increase as the problem gets larger. To study this, we determine, for various problem sizes, the minimally required n to solve the problem. To do so, we perform a bisection search in which, starting from $n = 1$, n is doubled until the problem is solved. Subsequently, a binary search is performed in the range between the last two tested values for n. Because this process is still subject to noise, rooted in the stochastic nature of EAs, we perform 10 independent bisection searches for every problem size (and in case of weighted MAXCUT, for every problem instance).

4.3 Results

In Figure 4, the minimally required population size and the associated number of evaluations (upon convergence) are shown. The outcomes of the 10 independent bisection searches are shown for each problem size (for each instance in case of weighted MAXCUT). A least-squares polynomial fit of the form $\alpha \cdot l^{\beta}$ is also shown except for solving MAXCUT to optimality as it doesn't fit the data well. Instead, the lines we show connect the average values for each problem size.

All variants are virtually indistinguishable on onemax because the LT always contains every variable in a singleton set. However, close examination shows that the variant that uses H scales worst. These results are clearer on Trap 5 where only MI and NVI are virtually indistinguishable. Most clear are the results on NK-S1 where MI and NVI are again virtually indistinguishable but H

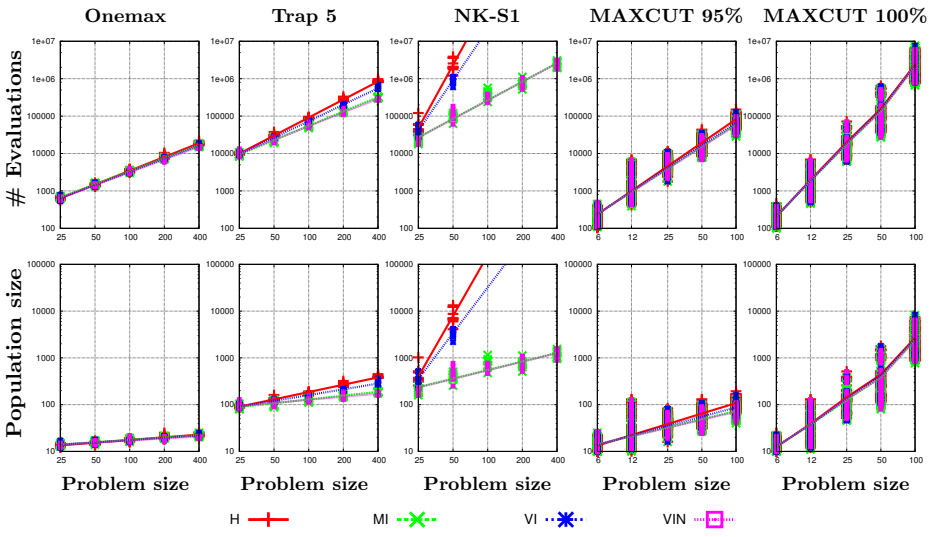


Fig. 4. Scale-up of LTGA on all problems, using different measures to build the LT

and VI clearly guide LTGA less efficiently. Combined with our analysis in Section 3, these results suggest that using LTGA on NK-S1 some variables converge much faster than others. Once converged, VI and, even more so, H favor these variables much stronger. Not only is this inefficient with respect to finding and mixing building blocks amongst variables where diversity still remains, it also creates strongly unbalanced trees with many large clusters, further reducing the efficiency of optimal mixing. MI and NVI are not affected by this cascading effect of converging variables. Finally, it appears that for solving our randomly generated weighted MAXCUT instances proper linkage learning is not as crucial because just like on onemax, all variants scale quite similarly. Differences do seem to increase with problem size, although far less severly. Proper linkage learning may still be required when solving instances with specific structure, especially when targeting the global optimum. A more in-depth study on weighted MAXCUT and the impact of linkage learning will be topic of future research.

As expected from Section 3, LTGA performs best when the measure used is MI or NVI (or, equivalently, MNI). For these two alternatives, using a Mann-Whitney U test at a significance level of 1%, results differ only for onemax with $l = 400$ (in favor of NVI), trap 5 with $l = 400$ (in favor of NVI), never for NK-S1, never for weighted MAXCUT 95% and only for one instance with $l = 100$ for weighted MAXCUT 100% (in favor of MI). Arguably therefore, it is virtually impossible to prefer one measure over another (based on the selected problems).

5 Conclusions

The linkage tree genetic algorithm (LTGA) was previously combined with different measures for building its linkage model (the linkage tree). In this paper

we took a closer look at these measures, related them to the convergence of an EA and we identified potential biases in the measures. The closest correspondence to the notion of a building block was found for the mutual information (MI) measure and a normalized (MNI) variant that is obtained by dividing by joint entropy. MNI is equivalent to the normalized variation of information (NVI) measure, even when using the less-computationally demanding pairwise measure adaptation known as UPGMA. The difference between MI and MNI/NVI is that the former has a slight preference for less converged variables. These measures only disagree in less than 4%, and when they do differ, these differences are very small. Consequently, LTGA was found to perform very similarly for these two measures on a set of three benchmark problems from linkage learning literature as well as on a combinatorial optimization problem: weighted MAXCUT. Only very few statistically significant differences could be found in the performance of LTGA using MI or NVI/MNI and even then the results were very close.

References

1. Bosman, P.A.N., Thierens, D.: Linkage neighbors, optimal mixing and forced improvements in genetic algorithms. In: Proc. of the Genetic and Evolutionary Computation Conf., GECCO 2012. ACM Press, New York (to appear, 2012)
2. Deb, K., Goldberg, D.E.: Sufficient conditions for arbitrary binary functions. Annals of Mathematics and Artificial Intelligence 10(4), 385–408 (1994)
3. Harik, G.R., Lobo, F.G., Sastry, K.: Linkage learning via probabilistic modeling in the extended compact genetic algorithm (ECGA). In: Pelikan, M., et al. (eds.) Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications, pp. 39–61. Springer, Berlin (2006)
4. Kraskov, A., Grassberger, P.: MIC: Mutual information based hierarchical clustering. In: Emmert-Streib, F., Dehmer, M. (eds.) Knowledge Incorporation in Evolutionary Computation, pp. 101–123. Springer, Berlin (2009)
5. Pelikan, M., Hauschild, M.W., Thierens, D.: Pairwise and problem-specific distance metrics in the linkage tree genetic algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2011, pp. 1005–1012. ACM Press, New York (2011)
6. Pelikan, M., Sastry, K., Goldberg, D.E., Butz, M.V., Hauschild, M.: Performance of evolutionary algorithms on NK landscapes with nearest neighbor interactions and tunable overlap. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2009, pp. 851–858. ACM Press, New York (2009)
7. Rendl, F., Rinaldi, G., Wiegele, A.: Solving Max-Cut to optimality by intersecting semidefinite and polyhedral relaxations. Math. Prog. 121(2), 307 (2010)
8. Rubinstein, R.Y.: Cross-entropy and rare events for maximal cut and partition problems. ACM Trans. on Modeling and Computer Simulation 12(1), 27–53 (2002)
9. Thierens, D.: The Linkage Tree Genetic Algorithm. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 264–273. Springer, Heidelberg (2010)
10. Thierens, D., Bosman, P.A.N.: Optimal mixing evolutionary algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2011, pp. 617–624. ACM Press, New York (2011)